Com S 336
Fall 2020
Notes for exam 2
Tuesday, November 24, 9:45 - 11:45 am

General information

This will be a 120-minute exam, administered on Canvas during our final exam period. Open notes, open books, but **no collaboration**. You can do internet searches for information, but you must pledge that you will not solicit help from any other person or online forum.

(Note: If you have an accommodation letter from Student Accessibility Services I'll contact you separately.)

Note

For coding problems you can use three.js or CS336Object. For CS336Object, assume we have functions **`drawSquare`** for rendering a 1 x 1 square, **`drawCube`** for rendering a 1 x 1 cube and **`drawSphere`** for a sphere of radius 1, using some globally defined camera and shader. For example, either code snippet below will render a sphere of radius 1 centered at 2, 3, 4:

```
var obj = new CS336Object(drawSphere);
obj.setPosition(2, 3, 4);
var worldMatrix = new THREE.Matrix4();
obj.render(worldMatrix);
```

```
var geom = THREE.SphereGeometry(1);
var mat = ... // Basic, Phong, etc
var obj = new THREE.Mesh(geom, mat);
obj.position.set(2, 3, 4);
var scene = new THREE.Scene();
scene.add(obj);
renderer.render(scene, camera);
```

Topics and sample questions

The exam is designed to cover the material since Exam 1, but it is inevitable that there is some dependence on earlier topics. You should probably review anything you had trouble with on Exam 1.

Indices

- *Given an array of vertices, write code to generate an array of indices that will render a **wireframe** of the model with the call* **`gl.drawElements(gl.LINES, numIndices, gl.UNSIGNED_SHORT, 0).`**
  Assume that the vertices are ordered so that the model could be rendered with a call to gl.drawArrays(gl.TRIANGLES, 0, numVertices).

Normal vectors

- Does a normal vector always have to be perpendicular to an associated triangle? Briefly explain.

*- Give an example in which the normal matrix is not equal to the view * model transformation*

- Given an array of vertices, write code to generate a parallel array of face normals. Assume that the vertices are ordered so that the model can be rendered with a call to gl.drawArrays(gl.TRIANGLES, 0, numVertices). That is, each group of 9 numbers represents the three vertices for one triangle, ordered counterclockwise. In the normals array, you'd have the face normal for that triangle (three numbers) appearing three times.

Lambert shading

- Why do we have to normalize L and N in our lighting calculation?

- Why does the shader code have to take the maximum of 0 and (L dot N) ?

- Why doesn't the diffuse lighting calculation need to use the direction of the view point?

Phong (ADS) lighting model

- Suppose at a given vertex, L points to the light and N is the normal vector and V points to the view point. Assume all three are normalized. ***Sketch*** *a picture and label the relevant vectors and angles. Write down the three components of the lighting model as accurately as you can.*

- What is the purpose of the exponent in the specular component? What changes if we make the exponent bigger?

- How does the Blinn-Phong model differ from the Phong model? Justify with a sketch that the angle between H and N is half the angle between R and V (when all vectors are in the same plane).

Gouraud and Phong shading

- Briefly explain the difference between Gouraud and Phong shading

- Explain the nature of the artifacts that occur using Gouraud shading with a specular component

- Why do we have to normalize after interpolating vectors? Give a concrete example showing that linearly interpolating between two vectors of length 1 does not necessarily give you a vector of length 1.

- We normally do lighting calculations in eye coordinates. What would be different/easier/harder if we wanted to do it in world coordinates?

- Starting with the vertex shader from Lighting2c.html, add a uniform variable representing the position of the camera, given in world coordinates. Then rewrite the shader so that the lighting calculation is done in world coordinates. ***Note: you can make the assumption that there is no nonuniform scaling, i.e., the given normal matrix (which you will no longer be able to use) is just view*model.***

- Same as above, but assume you ***don't*** *have a uniform variable for the camera position. (So the real question is, how do you get the camera's world-coordinate position if it's not given as a uniform variable? Recall that the view matrix looks like* $R^{-1}T^{-1}$ *where the columns of R are the camera's basis vectors and T is the translation to the camera's position; also recall that R is orthonormal. This makes it easy to recover T from the view matrix.)*

Hierarchy

- Suppose we have a 1 x 1 square whose position is represented as a three.js or CS336Object with the transformations:

scale: 2, 4, 1
translation: 0, 0, 0
rotation: 90 degrees counterclockwise about z

Then we add a child object with the transformations

scale 1, 1, 1
translation: 0, 5, 0
rotation: 45 degrees counterclockwise

Sketch what we are going to see when rendered, indicating dimensions.

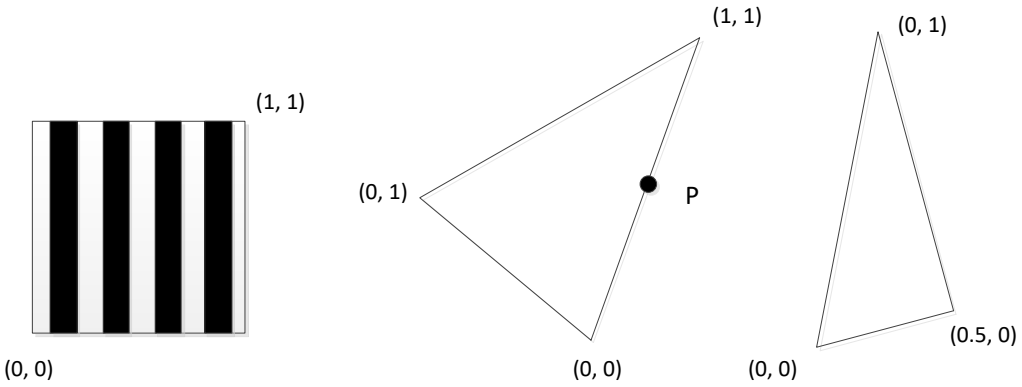
- Set up a planet rotating in a circular orbit around a fixed point (the "sun"), while also rotating on its vertical axis 365 times per orbit, where it axis of rotation is tilted 23.4 degrees from the vertical. Add a moon that orbits the planet at some smaller rate and is 1/10 the size of the planet. ***(Write actual Javascript code for creating the objects and setting up their relationships; you can use three.js objects or CS336 objects; show the animation loop, but you don't have to provide all details of a "draw" function for doing the rendering.)***

Texture mapping

*- Suppose you have the stripe texture shown below at left, and assume that for the polygons to the right, the given coordinates are the **texture coordinates** for those vertices.*

*a) What are the approximate texture coordinates of the interpolated point P? **Show** the location in the texture that will be sampled for a pixel at point P.*

*b) **Sketch** the appearance of the texture on both polygons.*



- Describe the problem of "magnification" in texture mapping. What artifacts result from it? What do we do to mitigate it?

- Describe the problem of "minification" in texture mapping. What artifacts result from it? What do we do to mitigate it?

- Why does WebGL require 2D textures to be square and to have a width and height that is a power of 2?

- Starting with the fragment shader from Lighting2c.html; suppose we have added a uniform variable `textureUnit` and a varying variable `fTexCoord` for an interpolated texture coordinate. Make the additional modifications that would be needed to do the following: sample from the texture, convert the color to a greyscale value by averaging the red, green, and blue components, and then use the resulting value to modulate the diffuse surface color.

- Suppose you have a fragment shader that samples from a texture and uses the sampled value for the surface color. The sampling parameters are set to REPEAT. Modify the shader to give the effect of shrinking the texture down by a factor of 4. (That is, if the texture image you have loaded is a 2x2 checkerboard, after you modify the shader it gives the effect of having an 8x8 checkerboard.)

- Starting with the fragment shader from Lighting2c.html, suppose we have added a varying variable **`fTexCoord`** *for an interpolated texture coordinate, but there is no texture unit or actual texture image.) Write code in the fragment shader to simulate the effect of a 2x2 checkerboard texture for the diffuse surface color.*

- Given a simple vertex/fragment shader pair, modify them to sample from a cube map and use the sampled value as the surface color. (There are no texture coordinates.)

- Same, but assume you also have a uniform variable **`textureUnit`** *in the fragment shader with the texture unit for a cube map. Modify the shaders to get the effect of reflecting the cube map on the surface of the model. You can assume you have model, view, and projection matrices in the vertex shader and that there is no nonuniform scaling of the model.*

Bump mapping

If you look at the file for a normal map in an image viewer, it appears mostly blue. Why?

Suppose a vertex and fragment shader are set up to use bump mapping. Instead of using the sampled value from the normal map as the N vector, you just use

```
vec3 N = vec3(0.0, 0.0, 1.0);
```

what will it look like?

FBOs

Suppose you have code to render an outdoor scene, and now you want want to render an interior room that has a window to the outdoor scene. Describe the steps needed to do it.

Describe the shadow mapping algorithm in words/pseudocode.